**Final Iteration**

HGCG :

* Tvisha Gangwani (trg2128)
* Jiawen Li (jl5303)
* Evan Ziebart (erz2109)
* Jiahong He (jh3863)

1. Added user story 4.8 and more unit tests
2. GitHub Repo Link:

<https://github.com/JiahongHe/Personal-Facial-Identification-System>

1. Since we already finished our pre-commits and post-commits in first iteration, we continuous working on the same CI.

Pre-commit : Personal-Facial-Identification-System/test/[pre\_commit.sh](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/pre_commit.sh)

Which would run the following:

* Style checker: use pylint to automatically run static analysis on all the .py files, and save the output to ./test/reports/[styleCheckReport.txt](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/reports/styleCheckReport.txt)

(we found that this contains a lot of false positives)

* Install all the required Python module
* Run [test\_main.sh](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/test_main.sh) that would run all the tests.

Post-commit CI : Personal-Facial-Identification-System/.travis.yml

Which would:

* Setup the environment (linux operating system, Python)
* Install required Python packages.
* Run [test\_main.sh](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/test_main.sh) that would run all the tests.

1. User stories:
   1. **Title**: [Administrator] Update user information in the system
      * **Actor(s)**: The System (Djingo)
      * **Description**:
        + As an administrator, I want an ability to update user data in the system, such as name, password, photo, and songs, so that I can control how the system treats a user and fix a user’s errors. The conditions of satisfaction are be able to change a user’s name, password, photo, song list in the system.
      * **Basic flow**:
        + The system successfully updated user information.
      * **Alternate flows**:
        + While the system updates user information, user’s name or password is not found or incorrect
        + While the system updates user information, user’s photo or user’s song list is not found or its format is incorrect

* 1. **Title**: [Administrator] Add new user to the system
     + **Actor(s)**: The System (Djingo)
     + **Description**:
       - As an administrator, I want the ability to add new users to the system, so that they can access it. Once a user is added, they will remain in the system even when it is turned off and on again
     + **Basic flow**:
       - The system successfully added user to the system.
     + **Alternate flows**:
       - While the system adds user to the system, user’s name or password or photo, or song list error. The system failed to add the new user to the system
       - Once restart the system, the newly added user’s information disappeared.
  2. **Title**: [Administrator] Remove a user from the system
     + **Actor(s)**: The System (Django)
     + **Description**:
       - As an administrator, I want to remove a user from the system, in order to prevent them from accessing the system or to save space. Once a user is removed, their information is deleted and will not return. Once removed, a user can no longer access the system
     + **Basic flow**:
       - The system successfully removed user from the system.
     + **Alternate flows**:
       - Once the user removed from the system, this user still can login to the system with his/her removed password to get the removed information.

* 1. **Title**: [User] System can identify user through camera
     + **Actor(s)**: The System (Django), User
     + **Description**:
       - As a user, I want the system to correctly identify me when I look into the camera, so that it can greet me and play my song. Once I look at the camera, the system can correctly recognized my face. Using facial recognition to get my personal preferences
     + **Basic flow**:
       - User can be successfully identified by the system. Then, the system grab information about this recognized user’s, and play his/her favourite song.
     + **Alternate flows**:
       - While the system identifies user through camera, user is not staying still, so that the camera cannot capture a clear image of user’s face.
       - While the system grabs relevant information about the user, the user is not exist, which the information is not found. Then, the system categorize this user as unknown.
       - While the system grabs relevant information about the user, this user’s song is not playable, which might be an audio file error.
  2. **Title**: [User] Protected personal information with secured password in the system
     + **Actor(s)**: The System (Django), User
     + **Description**:
       - As a user, I want my information in the system to be password protected, so that others cannot change my information to things I don’t want. I have an ability to login to the system using my password. Other users cannot access my information or update my preferences without knowing my password. My password is stored securely
     + **Basic flow**:
       - User’s personal information are secured in the system, which is protected by a secured password.
     + **Alternate flows**:
       - While the system protects user’s personal information, user failed to login to the system by his/her secured password
       - While the system protects user’s personal information, users can access and update other users’ personal information and preferences.
       - While the system protects user’s personal information, user’s password is not stored securely.
  3. **Title**: [User] Update user’s password
     + **Actor(s)**: The System (Django), Users
     + **Description**:
       - As a user, I want the ability to update my password in the system, so that I can change it into something I can easily remember. Once I have created my account with the system, I can be able to update my password whenever I want.
     + **Basic flow**:
       - User’s password can be successfully updated and stored into the system.
     + **Alternate flows**:
       - While the user updates his/her password, user failed to update the password, which still need to use the old password to login to the application.
  4. **Title**: [User] Update user’s preferences
     + **Actor(s)**: The System (Django), Users
     + **Description**:
       - As a user, I want to be able to update my preferences such as full name, photo, and songs, so that I can control how the system responds when I look at it. My conditions of satisfaction are be able to change my first and last name, photo, and song list in the system.
     + **Basic flow**:
       - User’s preferences can be successfully updated and stored into the system.
     + **Alternate flows**:
       - While the user updates his/her preferences, user failed to update any of the preferences.
  5. **Title**: [Administrator] Specify custom behavior for edge and error cases in the system, so that I can handle unexpected situations.
     + **Actor(s)**: The System (Django)
     + **Description**:

As an administrator I want to be able to specify a default noise and action when a person is not known to the system. I also want to be able to specify a default noise and action when a user’s song cannot be played

**Basic flow**:

* + - There is a default noise and action that plays when the person is not known to the system.
    - **Alternate flows**:
    - An unknown person is identified by the system and the system does nothing

1. Unit tests and tests for facial recognition:

Currently, our test suite includes unit tests for all of the user stories implemented. Certain parts of the system are not in need of testing, because they are provided pre-tested as a part of the Django framework.

Location of Test Suite:

* + - https://github.com/JiahongHe/Personal-Facial-Identification-System/tree/master/test

1. Measurement of branch coverage:

We used the coverage.py tool to measure our unit testing code coverage. The system is integrated into our Travis CI automatic build process. It is run during each of our unit tests and the .coverage files are collected into a single report which is printed by the tool. The repository also includes the output from a run of coverage html, so that the latest code coverage is easy to explore.

Currently, our code coverage stands at 65%. Looking more closely at the report, some of the uncovered lines are handling exceptions. However, some of the uncovered lines of code are in views.py and models.py files, which indicates certain parts of the code are not yet being tested for by our unit tests. So, more work will need to be done to bring our unit testing up to speed.